AMENDMENTS TO THE CLAIMS

1. (currently amended) A multiple-precision, multiply-and-add computer operation for ~~handling~~ multiplying together a first operand with a second operand, at least one of the first and second operands ~~operand~~ having more than one natural word, and then adding an addend operand to the product of the first and second operands to produce a final result that is written into a multiple-natural-word-containing result vector, the multiple-precision, multiply-and-add operation comprising:

the ~~a~~ first operand;

the ~~a~~ second operand;

the ~~an~~ addend operand;

the ~~a~~ result vector; and

for each natural word of the second operand,

a block of *multiply-and-add* instructions that multiply the natural word of the second operand by all natural words of the first operand and store results of the multiply-and-add instructions as intermediate results, the block of *multiply-and-add* instructions that multiply the first natural word of the second operand by all natural words of the first operand additionally adding a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand, the block of *multiply-and-add* instructions containing no write dependencies.

2. (original) The multiple-precision, multiply-and-add operation of claim 1 wherein each block of *multiply-and-add* instructions contains only *multiply-and-add* instructions.

3. (original) The multiple-precision, multiply-and-add operation of claim 1 wherein a block of *multiply-and-add* instructions may contain *add* instructions in addition to *multiply-and-add* instructions.

4. (currently amended) The multiple-precision, multiply-and-add operation of claim 1 further including:

a number of blocks of *add* instructions that add the intermediate results and any remaining natural words of the addend operand to produce a final result that is stored into the result vector ~~that contains a sum of the addend operand and a product of the first and second operands~~.

5. (original) The multiple-precision, multiply-and-add operation of claim 1 wherein at least one of the first operand, second operand, and addend operand is contained within two or more registers.

6. (original) The multiple-precision, multiply-and-add operation of claim 1 wherein at least one of the first operand, second operand, and addend operand is contained within two or more natural words in memory.

7. (original) The multiple-precision, multiply-and-add operation of claim 1 wherein the result vector is contained within two or more registers.

8. (original) The multiple-precision, multiply-and-add operation of claim 1 wherein the result vector is contained within two or more natural words in memory.

9. (original) The multiple-precision, multiply-and-add operation of claim 1 wherein, because there are no write dependencies in the blocks of *multiply-and-add* instructions, all *multiply-and-add* instructions of each block can be executed together in parallel.

10. (currently amended) A method, carried out by a computer, for multiplying a first operand by a second operand to produce an intermediate product to which an addend operand is added to produce a result stored in a result vector, at least one of the first operand, second operand, and addend operand having more than one natural word, the method comprising:

for each natural word of the second operand,

using a block of *multiply-and-add* instructions to multiply the natural word of the second operand by all natural words of the first operand and store results of the *multiply-and-add* instructions as intermediate results, when multiplying the first natural word of the second operand by all natural words of the first operand additionally adding a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand, the block of *multiply-and-add* instructions containing no write dependencies.

11. (original) The method of claim 10 wherein each block of *multiply-and-add* instructions contains only *multiply-and-add* instructions.

12. (original) The method of claim 10 wherein a block of *multiply-and-add* instructions may contain *add* instructions in addition to *multiply-and-add* instructions.

13. (currently amended) The method of claim 10 further including:

using a number of blocks of *add* instructions that add the intermediate results and any remaining natural words of the addend operand to produce a final result that is stored into the result vector that contains a sum of the addend operand and a product of the first and second operands.

14. (original) The method of claim 10 wherein at least one of the first operand, second operand, and addend operand is contained within two or more registers.

15. (original) The method of claim 10 wherein at least one of the first operand, second operand, and addend operand is contained within two or more natural words in memory.

16. (original) The method of claim 10 wherein the result vector is contained within two or more registers.

17. (original) The method of claim 10 wherein the result vector is contained within two or more natural words in memory.

18. (original) The method of claim 10 further including executing some or all of the *multiply-and-add* instructions of each block of *multiply-and-add* instructions in parallel.

19. (currently amended) A multiple-precision, multiply-and-add computer operation for ~~handling~~ multiplying together a first operand with a second operand, at least one of the first and second operands ~~operand~~ having more than one natural word, and then adding an addend operand to the product of the first and second operands to produce a final result that is written into a multiple-natural-word-containing result vector, the multiple-precision, multiply-and-add operation comprising:

the ~~a~~ first operand;

the ~~a~~ second operand;

the ~~an~~ addend operand;

the ~~a~~ result vector; and

for each natural word of the second operand,

a means for multiplying the natural word of the second operand by all natural words of the first operand and storing results as intermediate results, the means for multiplying the natural word of the second operand by all natural words of the first operand additionally adds a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand without write dependencies.